

Tinfoil Chat

Instant messaging with endpoint security

Markus Ottela
University of Helsinki, Finland

`oottela[at]cs.helsinki.fi`
DRAFT

1 Abstract

Tinfoil Chat (TFC) is a high assurance encryption system that operates on top of messaging clients. Built on free and open source hardware and software the secure by design implementation protects not only data in transit against passive and active attacks, but also the end points against CNE practiced by organized crime and TLAs such as the NSA, GCHQ and BKA.

1. Authenticated encryption uses either OTP and one-time MAC, or cascaded set of symmetric ciphers (Keccak-512-CTR, XSalsa20, Twofish-CTR and AES256-GCM) and set of authentication algorithms (GMAC, HMAC-SHA2-512 and SHA3-512 MAC).
2. Keys are generated by mixing `/dev/(u)random` with vN whitened, SHA3-512 compressed entropy, sampled from an open circuit design HWRNG.
3. Endpoints are secured by separating encryption and decryption on isolated TCB-devices, that interact with a networked computer through open circuit design data-diode enforced unidirectional gateways. Removal of bidirectional channels prevents exfiltration of keys and plaintexts even with exploits against zero-day vulnerabilities in software against OS of TCBs.
4. Trickle connection hides metadata about when and how much communication is taking place by sending a constant stream of encrypted data to receiving TCB units.

Contents

1 Abstract	1
2 Acknowledgements	3
3 Introduction	3
4 Background	3
4.1 Unencrypted IM services	4
4.2 Encrypted IM services	5
4.3 OTR and Axolotl, End-to-end-encrypted IM	6
5 TFC: Encrypted IM with endpoint security	7
6 Encryption	9
6.1 Preventing loopback prevents key exchange algorithms	9
6.2 One-time Pad	9
6.3 Cascading Encryption	9
7 Random number generation	10
8 Endpoint security	12
8.1 Enforcing unidirectional traffic	12
8.2 The data diode circuit diagrams	13
8.3 Perfboard prototype of the modified data diode	14
8.4 Testing the data diode performance	15
9 Security evaluation	16
9.1 Encryption scheme - TFC OTP	16
9.2 Pad reuse risk	17
9.3 Encryption scheme - TFC CEV	18
9.4 Replay attacks and existential forgery	20
9.5 Physical key compromise	20
9.6 Key entropy	21
9.7 Software vulnerabilities	21
9.8 Data remanence	21
9.9 Targeted attacks	22
10 Conclusion	23
11 Copyrights	23
12 Appendix A: Summary of TFC key features	24
12.1 TxM	24
12.2 RxM	24
12.3 NH	24
12.4 Other software	24
13 Appendix B: List of terms and abbreviations	25

2 Acknowledgements

We would like to thank Professor Douglas W. Jones from the University of Iowa for support in understanding the working principle of the data diode. We would also like to thank Juha and Markus Niinkoski, Ari Vainio, colleagues and numerous people in online communities: Their shared expertise and encouragement have brought the project further than was ever intended.

3 Introduction

In this paper we describe TFC suite on high level. We measure the quality of generated keys and evaluate the security of cipher configurations and overall construction against known passive, active and targeted remote surveillance techniques revealed to be used in state sponsored spying. Based on those we show how unidirectionally connected, split trusted computing base (TCB) makes unethical bulk-CNE (automated end point exploitation) infeasibly expensive.

In chapter four we present societal background on mass surveillance technology to help justify the approach chosen for TFC. In chapter five we introduce the device layout and make the security claim. Chapter six looks at generation of trustworthy encryption keys by sampling entropy from an open source HWRNG. In chapter seven we take an in-depth look at the working principle and security benefits the hardware data diode provides against remote data exfiltration. Finally, in chapter eight we make a thorough security self-audit on the implementation. We also apply the full-disclosure principle and dissect remaining vulnerabilities in TFC.

The source code is at beta stage. TFC has not yet been peer reviewed by a third party so users should proceed with caution. Despite extensive testing, bugs might cause problems with stability or cause data loss. The source code is available at Github:

<https://github.com/maqp/>

User manual that contains build instructions for HWRNG and data diode, along with instructions for downloading, verifying, installing and using TFC, can be downloaded from

<http://cs.helsinki.fi/u/oottela/tfc-manual.pdf>

4 Background

The global surveillance disclosure of 2013 started by NSA whistleblower Edward Snowden has revealed that the NSA, among other western intelligence agencies, is intercepting protected communications on worldwide scale.

Debate over the issue has pushed US government to reform the way information is collected, but so far it has only affected the bulk collection of domestic phone metadata. UDHR Article 12 guarantees all humans equal right to privacy but policy of the US is to divide individuals to those whose privacy is- and those whose privacy is not protected by the 4th amendment. This means data of non-US citizens (96% of internet users) is considered "*fair game*"; Their privacy depends solely on technological solutions.

Among the leaks on NSA's toolkit, implications that current IM applications are insecure have emerged. The breakdown on NSA-intercepted communications provided to Washington Post by Snowden indicates more than 75% of intercepts consists of instant messages.¹ It could thus be argued, addressing IM security has the highest priority.

¹<http://apps.washingtonpost.com/g/page/world/communication-breakdown/1153/>

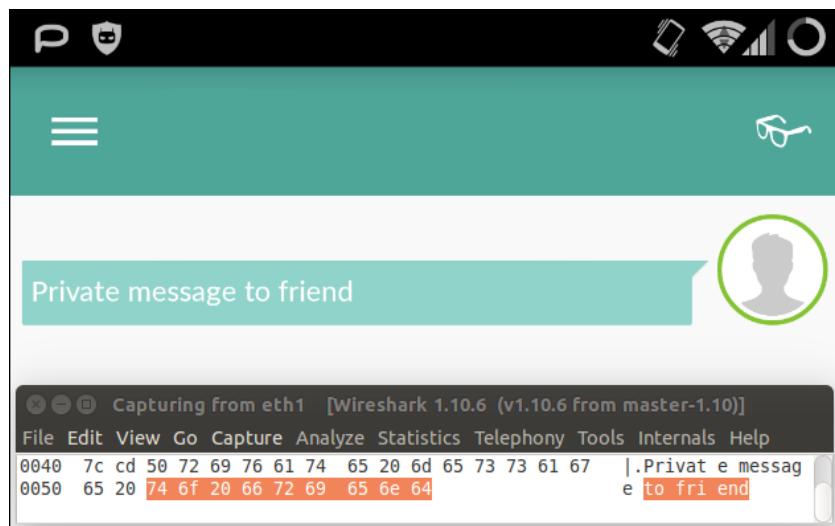
4.1 Unencrypted IM services

Unencrypted IM platforms such as ICQ, Palringo and X-Fire provide no confidentiality or integrity to messaging. Every entity that forwards the TCP packet including network administrators, ISPs, TLAs, and the staff in charge of IM server can undetectably read and change the content of any message.

Snowden documents have shed light to massive bulk collection programs of NSA and GCHQ. NSA's program Upstream is split under four codenames: FAIRVIEW and BLARNEY monitor the fibers of inland US, STORMBREW those that come ashore, and OAKSTAR fibers outside the US.² The packets crossing the US soil that involve a foreigner with at least 51% probability are intercepted by default.³ This is problematic, as significant amount of servers of global Internet companies are located in the US, and because packet routing uses the fastest channel available, and many times it is via the US.

Additionally, program RAMPART-A provides *"access to international communications from anywhere around the world"*⁴ and Executive Order 12333 allows NSA to intercept communications of US citizens from foreign interception points.⁵ In figure 1, a packet capture of Palringo messenger recorded with network protocol analyser Wireshark shows the confidential discussion can be intercepted by dragnet surveillance systems without complicated attacks. Data can be collected from choke-points and analysed off-site.

Figure 1: Palringo IM and Wireshark Pcap



²<http://electrospace.blogs.com/2014/01/slides-about-nsas-upstream-collection.htm>

³<http://www.bbc.com/news/technology-23051248>

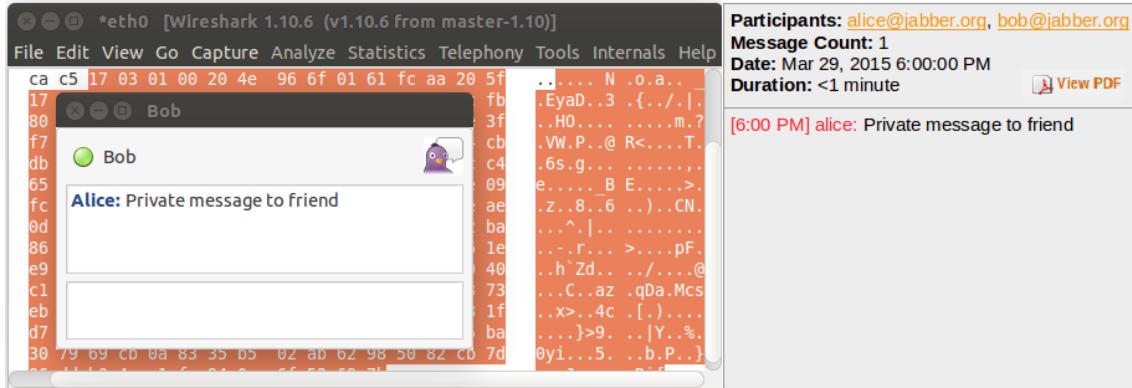
⁴<https://s3.amazonaws.com/s3.documentcloud.org/documents/1200864/tssinframpartaooverview-v1-0-redacted-information.pdf>

⁵<http://wapo.st/Ug0kLS>

4.2 Encrypted IM services

Today, most IM-platforms encrypt the connection between client and server with TLS. Such clients include Skype, Facebook messages, SnapChat, Telegram and Pidgin. The security of TLS depends on choice of key exchange, encryption and message authentication algorithms. Key length, entropy sources and defending against side-channel attacks also play critical role. The trustworthiness of client depends on availability of source code to verify the implementation correctly does what it claims to and nothing else; Proprietary products may be required by law to feature a capability for lawful interception.⁶

Figure 2: Pidgin IM, Wireshark Pcap and XMPP-server’s message monitoring system.



Because TLS-encryption is used only between client and server, the service provider has access to plaintexts, and it might store them either in compliance with data retention laws or to profit from data aggregation: surveillance is the business model of the Internet and breach of privacy law is many times seen by companies as a minor profit risk.⁷ Stored plaintexts can be acquired with court orders or with NSA’s PRISM program, that provides the TLA with direct access to servers of Microsoft, Facebook, Google and Apple.

Non-ephemeral TLS-encryption can be broken by obtaining private keys of companies with court orders accompanied with gag orders such as in the case of Lavabit.⁸ Alternatively, units such as NSA’s Tailored Access Operations (TAO) that specialize in CNE can exfiltrate keys from servers by exploiting unknown software vulnerabilities in server-side software.⁹ Theft of private RSA-keys makes retrospective decryption of traffic collected by Upstream trivial.

Forward secret, ephemeral DHE key exchange prevents retrospective decryption. However, theft of private DHE keys make transparent MITM invisible. This attack can be made easier with compelled certificate creation attacks.¹⁰ Additionally, NSA would appear to have certificate authority resources.¹¹ This poses a serious risk where TLAs in control of the keys can generate rogue certificates in real time.

NSA’s Bullrun¹² program is claimed to be able to break many forms of encryption including some IM protocols. Details are classified ECI but mention CNE, collaboration with other TLAs, HPC and advanced mathematical techniques. These capabilities have been estimated to include RC4, DHE-1024, RSA-1024 and SHA-1.

⁶http://www.theregister.co.uk/2011/06/29/microsoft_skype/

⁷ibtimes.co.uk/facebook-sued-selling-details-private-messages-advertisers-1430875

⁸<https://upload.wikimedia.org/wikipedia/commons/4/48/LavabitCourtDocuments.pdf>, page 36

⁹<http://threatpost.com/heartbleed-saga-escalates-with-real-attacks-stolen-private-keys/105436>

¹⁰<https://s3.amazonaws.com/files.cloudprivacy.net/ssl-mitm.pdf>

¹¹https://www.youtube.com/watch?v=oE92vJn_Ls8#t13m23s

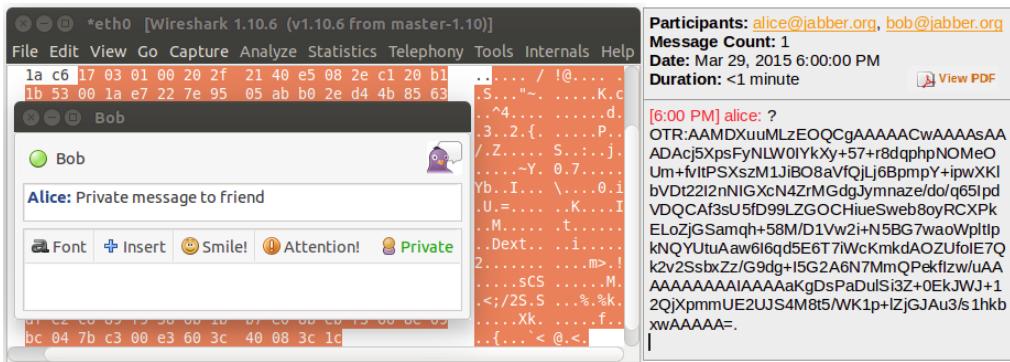
¹²<https://snowdenarchive.cjfe.org/greenstone/collect/snowden1/index/assoc/HASHc851/1420e5c8.dir/doc.pdf>

4.3 OTR and Axolotl, End-to-end-encrypted IM

Designed by Ian Goldberg and Nikita Borisov, OTR messaging reflects the values of cypherpunk movement that advocates use of strong cryptography to achieve social change. OTR provides forward secrecy to users by constantly generating new shared secrets via advertised DH-ratcheting. Axolotl by Trevor Perrin and Moxie Marlinspike takes this one step further by removing the need to advertise public values during DH-ratcheting: cyclic hashing of key refreshes it for each message.

By encrypting the message between end point devices, the intermediary server is unable to view the actual content of conversation. Both protocols encrypt messages between endpoints, thus the intermediary server is unable to view content of messages. Off-band verification of DH public-value fingerprints detects MITM attacks by malicious actors in the network.

Figure 3: Pidgin OTR UI and messages from perspective of XMPP-server



In an article¹³ The New York Times claims NSA has been able to “*Unlock private communication such as end-to-end encryption (that cannot be decrypted even by the messaging service) such as the one used in Adium*”. The technique how NSA is able to subvert OTR remains unknown – however, CNE would appear to be the most feasible attack vector.

Exploitation of end point devices

Leaks revealed that as data is collected via programs Upstream and PRISM, content is analysed by various programs, especially XKEYSCORE, that is able to detect PGP-encrypted emails and presumably also end-to-end encrypted IMs. The program also detects whether the client is exploitable by TAO.¹⁴ Targets that appear interesting might attacked with QUANTUM, that either does a man-on-the-side attack by competing with serverHello packet or a man-in-the-middle attack based on DNS-injection.¹⁵ Target connects to FoxAcid server that injects the target with an exploit. A modular payload called UNITEDRAKE delivered by the exploit then either logs keystrokes with plugin GROK or exfiltrates either log files or private keys with plugin SALVAGERABBIT.¹⁶

DH-ratcheting with verified public keys means MITM will only succeed if attacker is in possession of private keys of the user. In the case of OTR, exfiltration of private DSA key makes all future MITM attacks invisible. While in the case of Axolotl users are able to trace trust all the way to initial handshake, exfiltration of both the identity keys and root key makes MITM that immediately follows the exfiltration, also invisible.

¹³<http://www.nytimes.com/interactive/2013/09/05/us/unlocking-private-communications.html>

¹⁴<http://cryptome.org/2013/07/nsa-xkeyscore.pdf> pages 17 and 25

¹⁵<http://www.wired.com/2014/03/quantum/>

¹⁶<https://firstlook.org/theintercept/article/2014/03/12/nsa-plans-infect-millions-computers-malware>

5 TFC: Encrypted IM with endpoint security

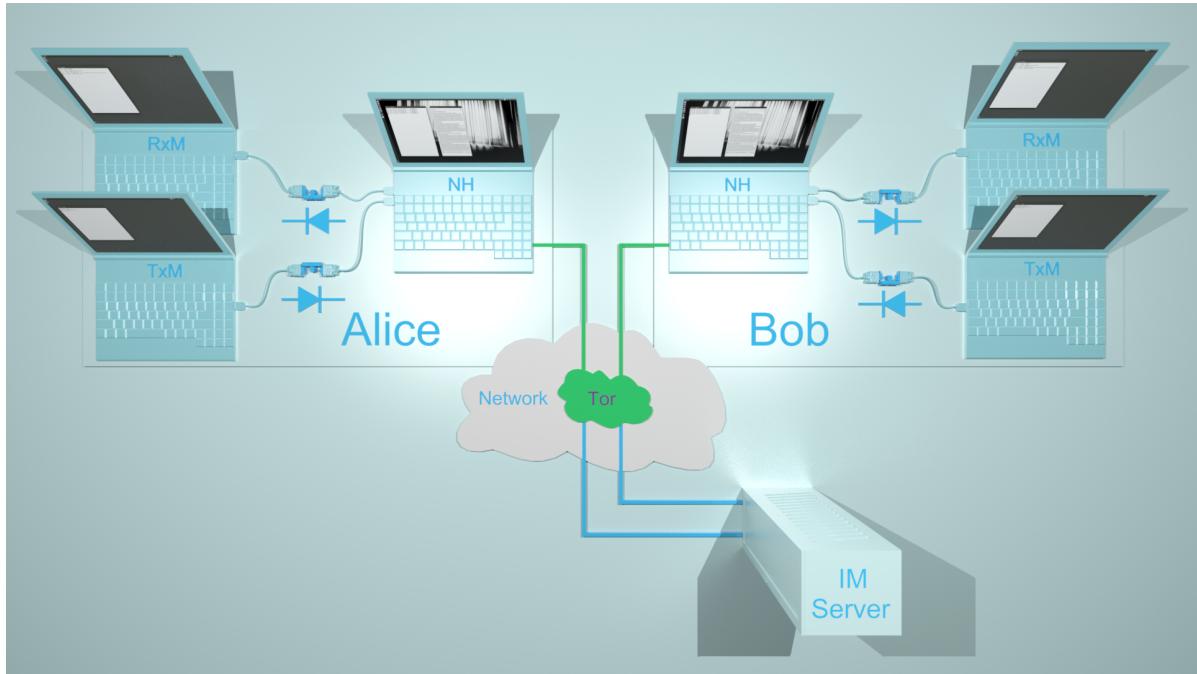
In TFC, Alice enters her message into Tx.py running on her Transmitter Module (TxM), a TCB separated from network. Tx.py encrypts the message and signs the ciphertext. TxM then relays the packet to Network Handler (NH) through RS-232 interface and a data diode.

The NH.py program on Alice's NH receives message from TxM via RS232 and forwards the message to IM client via dbus IPC. NH.py also sends a copy of the packet to her Receiver Module (RxM, another TCB separated from network) through RS-232 interface and a data diode, where the the ciphertext is authenticated, decrypted, displayed and optionally also logged.

IM clients of Alice and Bob connect to each other either directly, or through IM server. Both parties can optionally route their traffic through Tor-network.

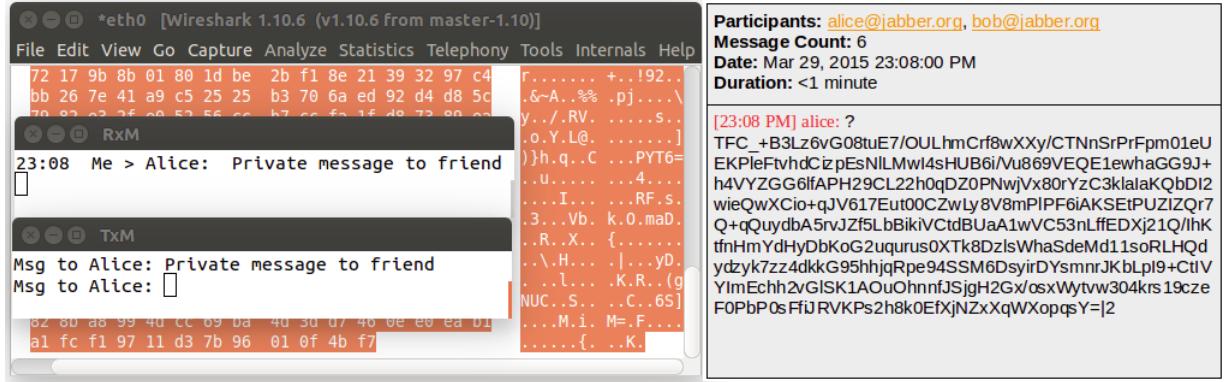
NH.py on Bob's NH receives Alice's packet from IM client via dbus and forwards it through RS-232 interface and another data diode to his RxM, where the ciphertext is authenticated, decrypted, displayed and optionally also logged. When the Bob responds, he will send the message using his TxM and in the end Alice reads the message from her RxM.

Figure 4: TFC device layout



Third party such as TLA or IM server only sees a Base64 representation of signed ciphertext, unless OTR is used to obfuscate TFC-protocol; Semantic analysis done by TLAs can easily detect use of TFC from messages that always start with the header '?TFC_'. However, since OTR messages are not effectively padded, traffic analysis can probably detect the static length of TFC messages anyway.

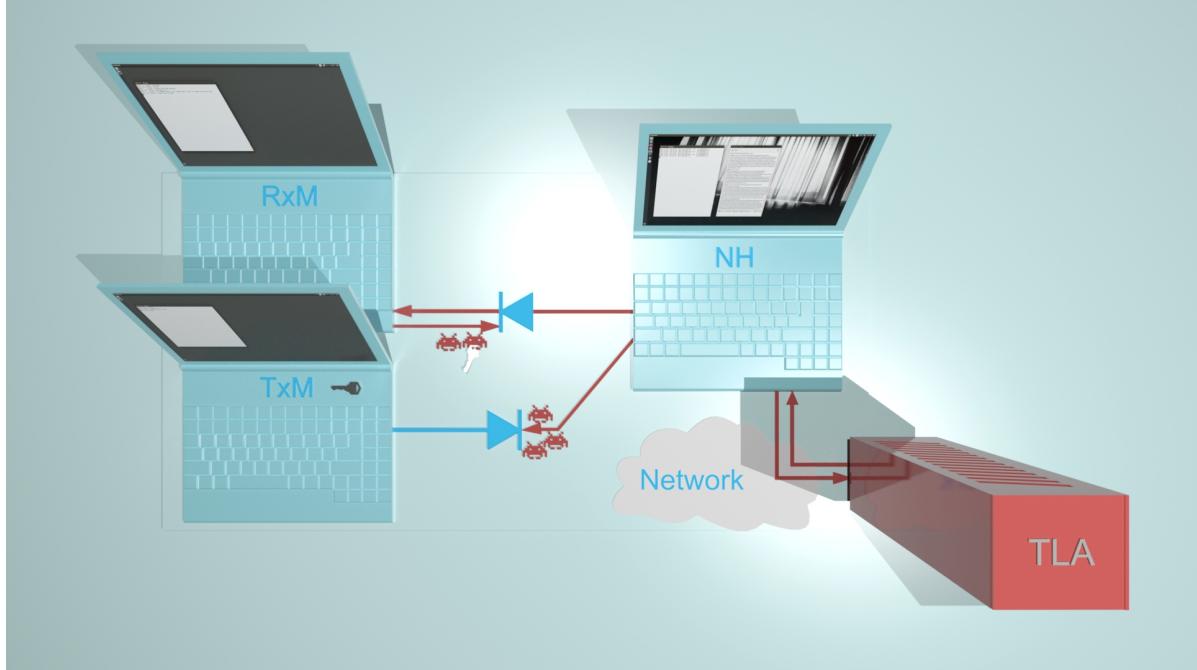
Figure 5: TFC software, Pcap and XMPP server's view of TFC message



End-point security of TFC is based on the fact no remote attacker / malware has bidirectional channel through which it could enter a TCB unit and exfiltrate sensitive keys and plaintexts:

- Malware never reaches TxM as data diode prevents inbound traffic.
- Malware that reaches RxM, can't exfiltrate keys as data diode prevents outbound traffic.
- The NH is assumed to be compromised, but it only handles ciphertexts and tags.

Figure 6: Security of TFC against key exfiltration with remote client-side attacks



6 Encryption

6.1 Preventing loopback prevents key exchange algorithms

To ensure no malware enters TxM, it is placed upstream in unidirectional data flow compared to NH and RxM. Automated intake channel for public RSA keys and DHE values from NH or RxM could infect TxM, and malware could then exfiltrate keys and plaintexts to attacker in network through TxM's data diode. Manual entry of long public RSA keys and DHE values to TxM is inconvenient, and the RxM can not be trusted to generate the shorter, shared secret. Thus, key exchange algorithms are not practical.

Unless the target is physically unreachable, a more practical solution is to generate a set of pre-shared symmetric keys on TxM that are then copied to both personal and recipient's RxM either through a high-speed data diode or a removable media that is destroyed after the procedure.

6.2 One-time Pad

NSA's recent policy statement to upgrade Suite B algorithms to quantum-safe cryptography might indicate that quantum computers are within reach of nation states some time in the near future.

TFC-OTP provides perfect secrecy and remains unbreakable regardless of advancements in cryptanalysis assuming specific conditions of using truly random pads, ensuring pads are never reused, and that the pad is destroyed immediately after use, are met.

Classic one-time MAC construction provides unconditionally secure message integrity. Additionally, overwriting of keys immediately after use provides TFC-OTP forward secrecy.

It is many times argued that if a secure channel for OTP-key transmission exists, users might as well use it for message transmission. However, by pre-sharing a lot of key material during inconvenient meeting, secure conversations can be continued over insecure but convenient channel for a long time. The drop in price of storage space means a \$5 microSD card can store 8GB of keyfiles to encrypt more than 18 million messages.

6.3 Cascading Encryption

Symmetric algorithms that use 256-bit keys are secure against Grover's quantum algorithm but they have not been proven secure. Based on the recommendations of the technical community including Edward Snowden¹⁷, TFC-CEV uses a set of symmetric ciphers with different internal structures. This way decryption would require cryptanalytic attacks against Keccak Sponge function, XSalsa20 ARX, Twofish Feistel Network and AES SP-nework.

Keccak-512-CTR uses a 512-bit key and other ciphers use 256 first bits of individual 512-bit keys, yielding a total of 1280 bits of symmetric security. Forward secrecy is obtained by cyclic hashing of keys though SHA3-512 immediately after use, similar to SCIMP.¹⁸ Authentication is handled by GMAC, HMAC-SHA2-512 and SHA3-512 MAC. The two latter are evaluated individually and they also authenticate the GMAC. HMAC-SHA2-512 uses individual 512-bit key and SHA3-512 MAC uses a 1144-bit key truncated from concatenation of three 512-bit individual keys.

Since MITM-attack free physical meeting is the preferred authentication method for many end-to-end-encrypted tools such as Signal, OTR, ChatSecure and Telegram, physical sharing of PSKs or OTP keyfiles does not add notable inconvenience to users.

¹⁷<https://www.youtube.com/watch?v=7Ui3tLbzIgQ#t=12m40s>

¹⁸<https://silentcircle.com/scimp-protocol>

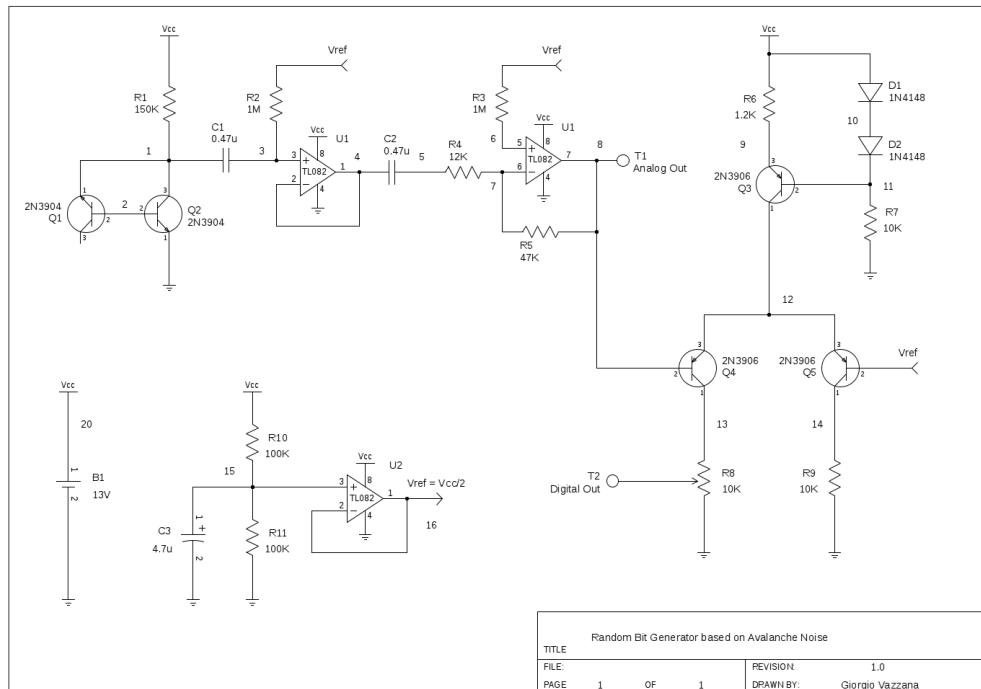
7 Random number generation

While strong randomness is important for both TFC-versions, TFC-OTP requires random bits at much larger quantities. The possibility that keys generated by processor come from low entropy pool can not be skated over. Commercially available peripheral devices for random bit generation also share the risk of undermined functionality. In many cases, the high price makes aftermarket solutions unavailable to the general audience. The author argues that best solution is an affordable, simple, open design HWRNG that can be built from off-the-shelf components inconspicuously and replaced in case breach of physical security is suspected. Instructions for building the HWRNG are available in the user manual.

HWRNG

The HWRNG used in this project is a design ¹⁹ by Giorgio Vazzana and costs only about \$30 to build excluding tools. Random noise is generated by following process: Avalanche noise of reverse biased transistor is passed through a buffer and it is then linearly amplified by an op-amp. The analog noise is converted to digital in a comparator and after this, regulated to the acceptable level of [-1.5, 1.5] volts in respect to virtual ground at 1.5V. Level of digital noise thus varies between 0V and 3V. The digital noise can then be sampled through GPIO of a Raspberry Pi by measuring the voltage level at fixed intervals. The recommended operating voltage of HWRNG is 15 volts.

Figure 7: Circuit diagram of HWRNG



¹⁹<http://holdenc.altervista.org/avalanche/>

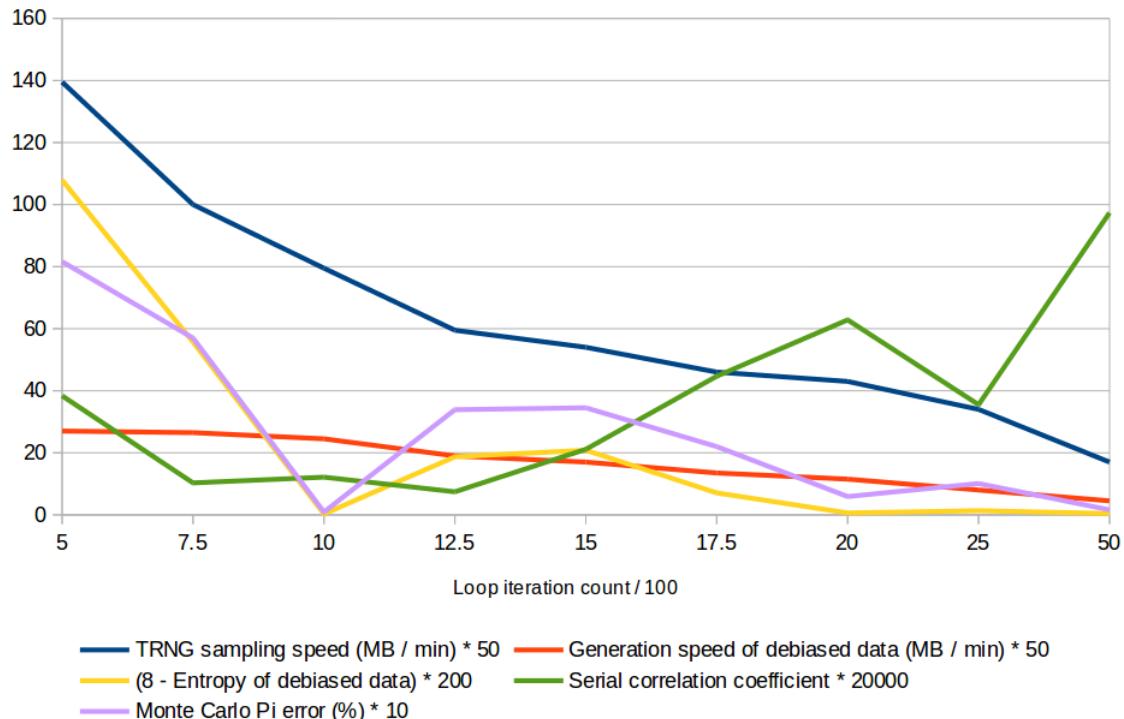
Key generation

7.0.1 OTP

The sample rate of HWRNG was adjusted with time consuming iterations of 'for loop' in range [500, 5000]. Thus, overclocking RPi might oversample the HWRNG producing bad entropy. Based on four independent samples at each speed, the average values were scaled to fit the chart (figure 8) plotting collection speed of sampled and von Neumann whitened entropy. With the value of 1000 iterations, the tool Ent measured the entropy of deskewed data to be very high (7.998697 bits / byte), so the value was chosen for TFC-OTP.

The output of HWRNG is \approx 8150 bytes / second which means that the system obtains key material for \approx 1100 messages every minute. Encryption keys are generated with genKey.py that runs getEntropy program as a subprocess to sample the HWRNG. After vN whitening, keys are optionally XORed with /dev/(u)random. As the mileage of every HWRNG varies, the genKey application also evaluates keyfile using tool Ent²⁰ and optionally also with Dieharder suite, that combines the Diehard battery of tests, NIST Statistical Test Suite and additional tests developed by Robert G. Brown.

Figure 8: Sampling speed chart



7.0.2 CEV

TFC CEV uses a static 2000 Hz sampling rate. genKey.py first loads 1 500 000 (43% headroom in case vN removes more than 3/4 of entropy) samples from HWRNG with getEntropy program. The sampled bits are then vN whitened thrice and between each pass, the input is compressed using SHA3-512 hash function (2:1 compress ratio). The entropy is then optionally XORed with /dev/(u)random and finally, before the eight individual 512-bit keys are stored, they are hashed together with separate keyboard inputs entered by the user, using SHA3-512.

²⁰<http://www.fourmilab.ch/random/>

8 Endpoint security

Quoting Edward Snowden, “*Encryption works. Properly implemented strong crypto systems are one of the few things you can rely on. Unfortunately, endpoint security is so terrifically weak that NSA can frequently find ways around it.*”²¹

To counter exfiltration attacks, encryption and decryption must be moved to a separate machine without network connectivity. Airgap is not a viable solution as transmission medium would provide a back channel for key/plaintext extracting malware and because such setup would be impractical for instant messaging purposes.

Establishing data link would provide the speed required but it would also provide a back channel once user wants to reply to a received message. The back channel can be eliminated if receiving and sending of messages is handled by separate computers. This type of hardware configuration prevents any third party from remotely at the same time inserting malware and exfiltrating confidential data such as keys and plaintexts from the TCBs (as described in chapter 5).

8.1 Enforcing unidirectional traffic

Removing the cable from rx pin of TxM’s serial port is not enough: any galvanic connection pair between two computers can in theory transmit information to both directions if firmware allows it by (malicious) design. This is why direction of data flow must be enforced. A paper²² by Douglas W. Jones describes an RS-232 data diode that utilizes LEDs and phototransistors: An electrically isolated solution that is supported by the Raspberry Pi (An RS-232 voltage shifter add-on card is required). To ensure no galvanic connections remain, TxM and RxM are required to run on batteries. The data diode works as follows:

On the transmitter side, high and low signal represented by alternating electrical current between Tx and GND pins switches two LEDs, connected in parallel with opposite polarities, on and off.

On the receiver side, two photo transistors move current between a voltage source and the Rx pin in alternating directions, depending on the state of the LEDs, reproducing the signal.

The optical elements lack the ability to reverse signal direction: unidirectional transmission is thus enforced on OSI layer 1.

Altering the schematics by Professor Jones, the LED - phototransistor assembly was replaced with an optocoupler that supports higher baud rates and is less prone to transfer errors.

Professor Jones kindly pointed out that a chip’s functionality may not be what the user expects. Hidden logic inside the IC design of an optocoupler might allow data to flow in reverse direction. Therefore use of two-terminal devices - LEDs and phototransistors - is much more secure a solution.

The data diode is a patented invention²³ in the United States until the end of 2017: it may be used freely in other countries.

²¹<http://www.theguardian.com/world/2013/jun/17/edward-snowden-nsa-files-whistleblower>

²²<http://homepage.cs.uiowa.edu/~jones/voting/diode/RS232tech.pdf>

²³<https://www.google.com/patents/US5703562>

8.2 The data diode circuit diagrams

Figure 9: The original data diode by Douglas Jones

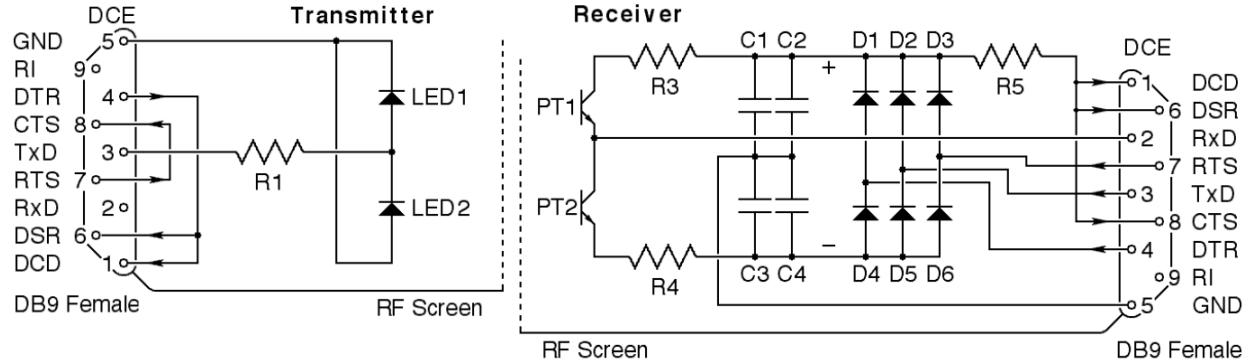
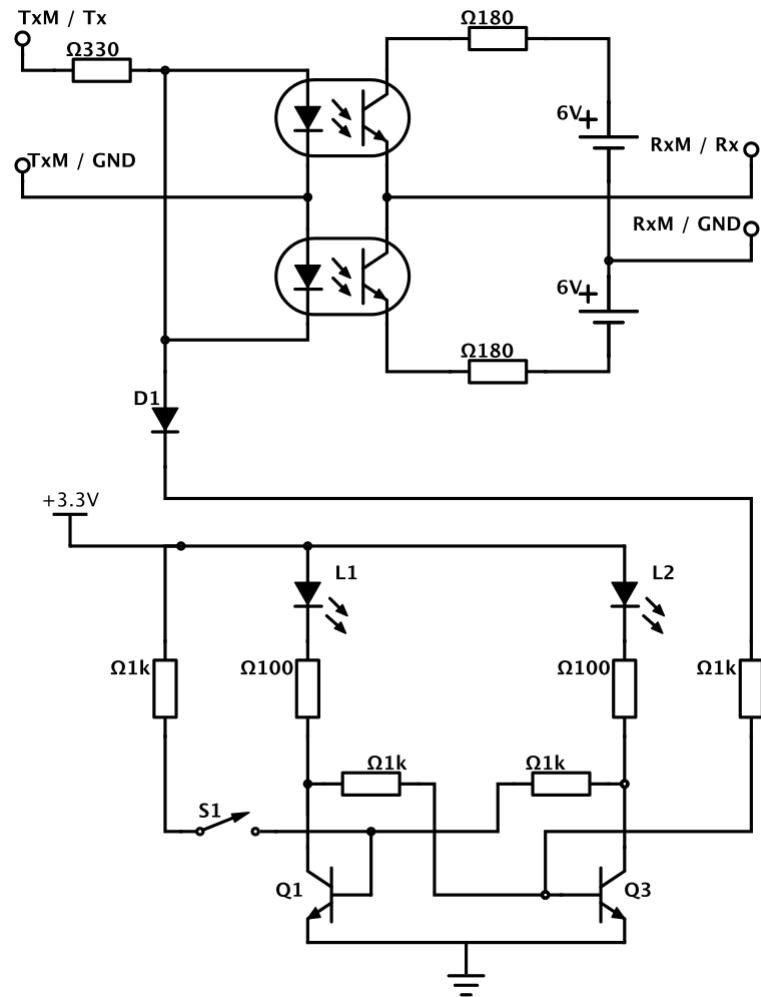


Figure 10: Modified data diode with flip-flop memory circuit.

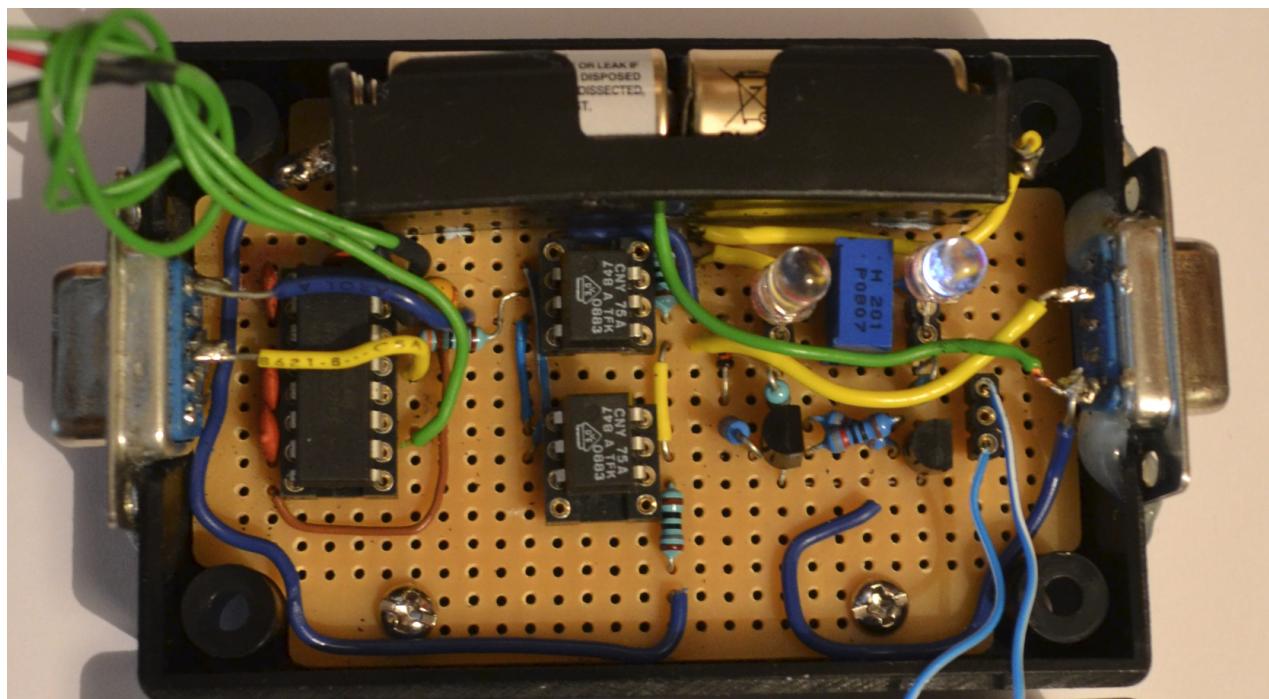


8.3 Perfboard prototype of the modified data diode

Step-by-step instructions for building the point-to-point version of data diode are provided in the user manual.

1. Optocouplers used in this data diode are the CNY75A by Vishay.
2. The 16-pin IC is the MAX3232 tranceiver that converts TTL level signals of RPi to RS232.
3. Voltage source is a pair of type 476A 6V batteries. Between the batteries is a metal plate that connects to the the receiver side's GND pin (green wire on right).
4. On right side of the optocouplers is the flip-flop circuit that detects covert data transmission during off-hours.
5. The shielding loop used in the original design was left out. In a simple test, emanations were unable to transmit data in reverse direction. Software of TFC is designed so that all confidential information the data diode transmits is encrypted. Thus, the main concern of EMSEC are the cables of display unit and keyboard.

Figure 11: The Data diode prototype



8.4 Testing the data diode performance

Continuously reproducing a single character "n" with varying baud-rates (no parity, 1 bit long stop), an oscilloscope graphed the signal from the data diode's transmitter and receiver side. The reproduced signal appears to match the original signal very well and thus should not cause problems in use. Increasing the baud-rate makes performance limits of the optocoupler apparent:

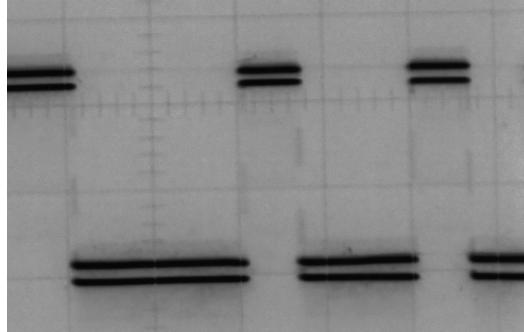


Figure 12: 1200 baud/s

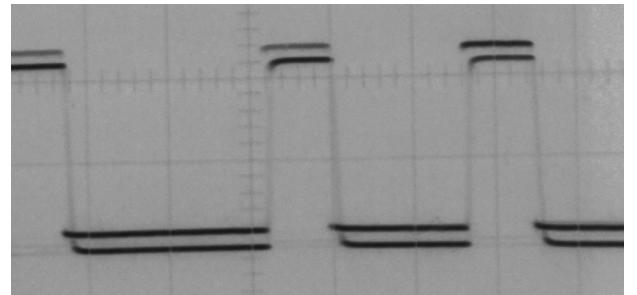


Figure 13: 9600 baud/s

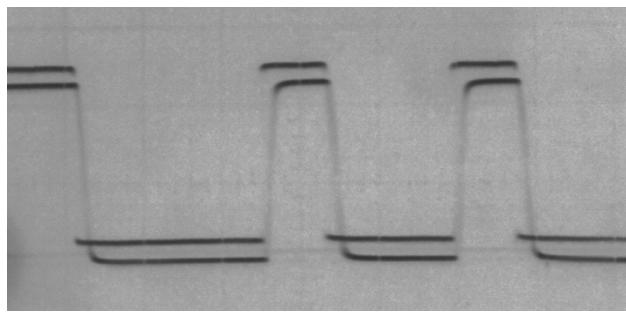


Figure 14: 19200 baud/s

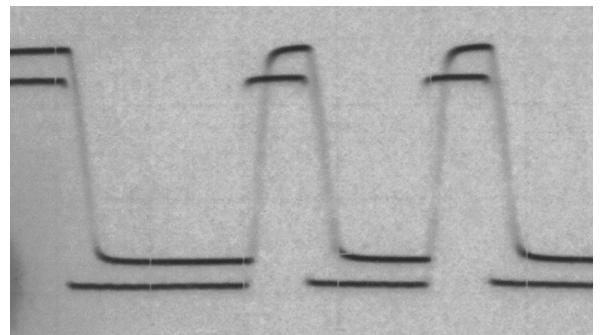


Figure 15: 38400 baud/s

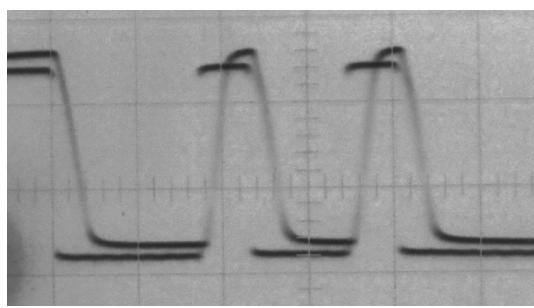


Figure 16: 57600 baud/s

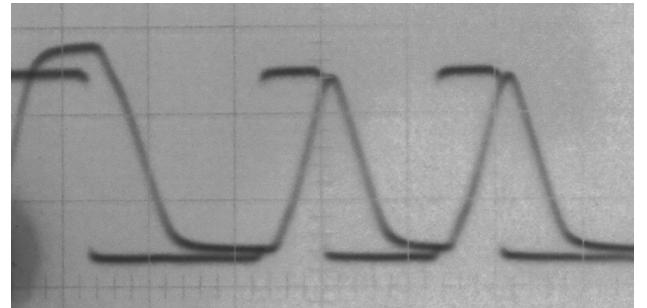


Figure 17: 115200 baud/s

9 Security evaluation

9.1 Encryption scheme - TFC OTP

9.1.1 Implementation

The encryption process of TFC-OTP is as follows:

$$\overbrace{k_i \oplus (M \parallel P)}^{\text{ciphertext}} \parallel \overbrace{c[L]k_{i+1}^L + c[L-1]k_{i+1}^{L-1} + \dots + c[1]k_{i+1} + k_{i+2} \pmod{q}}^{\text{MAC}} \parallel | \parallel i$$

k_i	is the OTP encryption key read from offset $l * i$ where l is message length.
\oplus	is the boolean exclusive-or operation
M	is the plaintext message
\parallel	denotes concatenation
P	is the padding that ensures messages have static length
$c[L]$	is integer representation of 64-byte ciphertext block
$k_{i+\{1,2\}}$	is a one-time MAC key calculated $k_{i+1} \pmod{q}$
q	is the M521, 13th Mersenne prime with 521 bit length
$ $	is a static character that separates keyfile line
i	is the offset of key K_i in keyfile

9.1.2 Ciphertext-only attacks

As presented by Dan Boneh ²⁴ : Let E be the cipher $c = m \oplus k$ over $\{K, M, C\}$ where K is the key space, M is the message space and C is the ciphertext space and let $|K| = |M| = |C|$.

$$\forall m, c : \#\{k \in K : E(k, m) = c\} = 1 \Rightarrow$$

$$\forall m, c : P(E(k, m) = c) = 1 / |K| \Rightarrow$$

$$\forall m_0, m_1 \in M \text{ where } \text{len}(m_0) = \text{len}(m_1) \text{ and } \forall c \in C :$$

$$P(E(k, m_0) = c) = P(E(k, m_1) = c), \text{ where } k \xleftarrow{r} K$$

Since any plaintext m_n encrypts to c with equal probability, no information about from which plaintext the ciphertext originates from can be extracted. Thus, no ciphertext-only attack exists for OTP.

²⁴<https://www.coursera.org/course/crypto>

9.1.3 Known-plaintext attacks

OTP encryption is malleable: if the attacker knows the plaintext, he can then XOR it with the ciphertext to recover used key, that can then be used to encrypt the tampered message.

To keep tampering undetected, the attacker would also need to successfully forge the MAC. By creating a polynomial of degree L for ciphertext with L blocks, and evaluating it at k_{i+1} , a unique random value that corresponds to the ciphertext is obtained. Another value k_{i+2} is obtained through uniform sampling from key space $K \in [0, q]$ (where q is a prime larger than the block size). When the two values are added together and reduced modulo q , the modulus is an irreversible value that gives the attacker no information about which keys were used to produce the MAC. This means MAC is unbreakable even with infinite computational power. The attacker's advantage in producing an existential forgery is same as guessing, $\frac{1}{q} = \frac{1}{2^{521}}$ which is negligible.

9.2 Pad reuse risk

If the keys are reused, attacker can correlate messages that use the same key simply by looking at the line number at the end of a message. Cryptanalysis such as crib-dragging is not always necessary: TFC's single letter padding may lead to catastrophic key exposure if keys are reused. The worst case scenario is, one of the messages was only one letter long. If the attacker XORs the ciphertext with the padding letter, he will end up with a key that can then decrypt all but the first letter of the longer message. By design, keys are never reused and thus, the attack vector is not present. TFC also features checks that prevent accidental reuse of keys.

9.2.1 Pad reuse protections

Tx.py checks the offset of every used key is higher than the one previously used and stores it inside the txc.tfc CSV-database. Tx.py then overwrites used keys thrice with /dev/urandom and once with ASCII 0x21 characters and verifies successful overwrite before proceeding. Loading of overwritten keys is also prevented. When changing keyfiles with /newkf command, old keyfiles are overwritten by tool shred (part of GNU coreutils) thrice and zeroed before removing.

To prevent accidental loading of a copy of old keyfile, Tx.py also features a boolean setting `check_key_hashes` that writes the SHA3-512 digest of every used key into global blacklist file 'usedKeys.tfc': every time a new key is loaded, it's hash is checked against the blacklist. Over time this setting may slow down Tx.py that is running on Raspberry Pi, so it's disabled by default. If you're using laptop computer as TxM, enabling the option shouldn't be a problem. The downside of enabling `check_key_hashes` is that adversary who physically compromises system can check whether intercepted ciphertext c was message m by checking whether a Keccak hash $H(c \oplus (m\|P))$ corresponds to a hash inside the file.

9.3 Encryption scheme - TFC CEV

9.3.1 Implementation

The encryption process of TFC-CEV is as follows:

$$(k_1 \oplus k_2 \oplus k_3 \oplus k_4 \oplus (M\|P)) \parallel | \parallel i$$

where

- k_1 is the Keccak-512-CTR keystream
- k_2 is the XSalsa20 keystream
- k_3 is the Twofish-CTR keystream
- k_4 is the AES-CTR keystream (GCM)
- M is the message
- P is the padding that ensures messages have static length
- $|$ is a static character that separates keyfile line
- i is the offset of key K_i in keyfile
- \parallel denotes concatenation

9.3.2 Ciphers

The substitution-permutation network based AES was selected over more secure Serpent as no free implementation for Python was found. The library used is pycrypto by Dwayne Litzenberger. GCM mode AES was selected as the outermost layer of encryption, as it provides integrity for the ciphertext. GCM-mode AES allows encryption of up to 2^{39} bits of plaintext for each IV-key pair. The standard length for plaintext is 2432 bits. The AES implementation uses a 256-bit symmetric key K_{i+3} together with 512-bit IV that is loaded from /dev/urandom.

Twofish is based on Feistel network. The library provided by Keybase is in ECB mode, so CTR-mode implementation was written for TFC-CEV: Plaintext is padded to 16 byte blocks. Next, a 128-bit nonce is loaded from /dev/urandom. For each plaintext block, the nonce is XORed with an increasing counter to create an IV, that is then encrypted with Twofish using the 256-bit symmetric key k_{i+2} to produce a keystream block. Once the keystream is as long as the plaintext, the two are XORed together to produce the ciphertext. Finally, the nonce is prepended to the ciphertext. The difference to standard version is, separate space for counter is not reserved: this allows the IV more entropy compared to concatenated 64-bit nonce and 64-bit counters.

The XSalsa20 is a stream cipher based on add-rotate-XOR (ARX). It is used with a /dev/urandom spawned, 192-bit nonce (length specified in libsodium). The independent, symmetric, 256-bit key k_{i+1} is used together with nonce to produce a keystream, which is then XORed with the plaintext to produce the ciphertext. In last step the nonce is prepended to the ciphertext.

The used Keccak-library is the official version written by Renaud Bauvin. The PRF is used in CTR mode: Plaintext is concatenated with padding to fill 512-bit blocks. The 512-bit symmetric key k_i is concatenated with 512-bit nonce loaded from /dev/urandom to produce a 1024-bit IV. For each block of plaintext, a matching length digest is appended to keystream. The next digest is generated by hashing the most recent digest prepended with the key. Once the keystream is as long as the plaintext, the two are XORed together to produce the ciphertext. In last step, the nonce is prepended to the ciphertext.

9.3.3 Security of cascading encryption

Assume the adversary has completely broken three out of four ciphers. The three broken ciphers can then be assumed to provide a predictable additive keystream (thus, the entropy added to plaintext is zero). During encryption this would equal the situation where instead of broken ciphers the plaintext would've been XORed with zero-entropy predictable string of zeroes (or ones). The resulting ciphertext bits for plaintext bits would then be:

$$\begin{aligned} 1 \oplus (0 \oplus 0 \oplus 0) &= 1 \oplus 0 = 1 \\ 0 \oplus (0 \oplus 0 \oplus 0) &= 0 \oplus 0 = 0 \end{aligned}$$

As the resulting ciphertext equals the plaintext, the encryption with the fourth, computationally secure cipher equals the situation where the three bad ciphers are not used at all. If the keys are independent, the three broken ciphers do not provide information about the keys or plaintext to adversary. A more thorough security proof is also available²⁵.

9.3.4 Authentication

Along GMAC, TFC-CEV uses HMAC-SHA2-512 and SHA3-512 MAC to verify cascaded ciphertext and GMAC. These two MACs are verified independently using a constant time function. All three MAC verifications need to succeed before the decryption takes place.

$$HMAC = H((k_5 \oplus opad \parallel H(k_5 \oplus ipad) \parallel ct \parallel GMAC))$$

where

H	is the SHA2-512 hash function
k_5	is an individual 512-bit key
ct	is the cascaded ciphertext
$GMAC$	is the AES-GCM MAC
\parallel	denotes concatenation
\oplus	denotes exclusive or / XOR
$ipad, opad$	are static paddings with large hamming distance

SHA-3 is immune against length extension attacks, thus the nested HMAC construction is not needed. However, According to paper²⁶ by Mostafa Taha and Patrick Schaumont, the optimal length of Keccak MAC-key is as close to $(2 * rate - 1)$ as possible but not over the value. In the case of Keccak-512, the optimal key size is $\lfloor (2 * 576 - 1) \rfloor = 143$ bytes.

$$MAC = H(\lfloor k_5 \parallel k_6 \parallel k_7 \rfloor \parallel ct \parallel GMAC)$$

where

H	is the SHA3-512 hash function
$k_{5..6}$	are three independent 512-bit keys
ct	is the cascaded ciphertext
$GMAC$	is the AES-GCM MAC
$\lfloor k \rfloor$	denotes truncation of key to 143 bytes
\parallel	denotes concatenation

²⁵<http://www.isiweb.ee.ethz.ch/papers/arch/umaure-mass-inspec-1993-1.pdf>

²⁶<http://rijndael.ece.vt.edu/schaum/papers/2013hosta.pdf>

9.3.5 Rubber-hose cryptanalysis

With both versions the keys are plausibly complex and change constantly, thus users can not be assumed to remember them. Because keys are overwritten as they are used, user is unable to provide old keys to any third party. Logging of messages is disabled by default and users should keep it that way to protect confidentiality of messages. Most effective tool to protect physical device is geological anonymity, best achieved by using Tor²⁷ and if possible, random public Wi-Fi access point. Protocol obfuscation by tunnelling TFC through OTR protocol further helps users to blend in.

Communication habits and time frames leak metadata about use purpose, time zones of users etc. TFC features a trickle connection where TxM constantly transmits to receiving TCB devices. Not even NH knows what packets contain actual messages or commands. Thus even if end point security of NH would be compromised, no information about content of packet could be obtained. The trickle connection or transmission of files / long messages can also be obfuscated to limited extent with random length sleep intervals specified by Tx.py's boolean setting "random_sleep". Some minimum sleep time might be needed to prevent flooding the XMPP server.

Hashes of OTR public keys are secure to be sent through TFC. Although OTR is not perfectly secure as discussed in 4.3, compromise of OTR only compromises the obfuscation part, not message content.

After initialization of OTR session, Alice and Bob should authenticate the user behind contact's hardware with a one-time secret question only the known participant knows the answer to: No encryption protocol prevents attacker from initializing conversation with Bob after capturing unlucky Alice.

9.4 Replay attacks and existential forgery

Rx.py automatically drops packets that repeat old key IDs. Tampered and crafted malicious packets also fail as MAC fails; RxM has no way to notify NH about MAC failure but constant time comparison is used to avoid reasoning by Legos.

By default, the Rx.py creates an event log for malformed/tampered or replayed packets.

9.5 Physical key compromise

When Rx.py receives a message, it overwrites keys that are related to it and any older keys of dropped packets. However, if Alice would send messages to Bob and their communications would be cut indefinitely, Bob's computer would continue to store decryption keys, that an adversary could later physically compromise. Best way to minimize the risk is to have a turn-based conversation: The recipient should send an acknowledgement as once the long message has been received. That way the sender knows the decryption key of long message has been overwritten.

²⁷<http://www.theguardian.com/world/interactive/2013/oct/04/tor-stinks-nsa-presentation-document>

9.6 Key entropy

Since Von Neumann's whitening algorithm only accepts digital signal's evenly common events of rising and falling edges, the bias of HWRNG doesn't affect the binomial distribution of finished entropy file. The Von Neumann algorithm also takes care of possible drift in bias over time and any bad spots that might occur during sampling.

Sample rate is set low enough not to induce short-term auto-correlation so while sampling from HWRNG is not a Bernoulli process, the deskewed entropy has negligible bias. XOR operation between debiased HWRNG entropy and uniformly distributed pseudo-random output of `/dev/(u)random` further improves the uniformity of keyfile without lowering the entropy.

The capacitor C3 of HWRNG should filter out noise from supply rails that might affect the bias but use of 15 volt battery is recommended. This also prevents side-channel leaks that evaluate power consumption of HWRNG: If constant power supply is the only option, power draw can be obfuscated by connecting high-power resistor ($5\ \Omega$) in parallel with the HWRNG.

In TFC-CEV, `genKey.py` allows user to mix in arbitrary length string of entropy from keyboard, that will then be hashed together with entropy from TFC using SHA3-512.

9.7 Software vulnerabilities

Unidirectionally connected physical devices makes TFC secure only against key and plaintext exfiltration with exploits against software vulnerabilities. TFC is **not** secure against malware that exploits a hardware covert channel (e.g. RF/Thermal/optical/sonic) between one of the TCB devices and any other device that provides networking capability (nearby smartphone / NH / other networked computer). Exploits similar to AirHopper, BitWhisperer and GSMem can be automated but they require higher level of targeting to ensure exploit works in user's hardware configuration.

TFC can not protect TxM during setup so if the system is compromised during this brief window of opportunity, all future keys generated can be exfiltrated through standard data diode channel. However, malware on TxM isn't able to tell what's on the other side of data diode – if it's the NH, or a spectrum analyser that does FFT calculations from the regenerated signal. The covert channel can be detected, if the period of audit lasts longer than malware remains dormant.

TFC can not protect RxM against attacks or malware that aim to disrupt the system and/or cause loss of data inside RxM. Compromised RxM can also change the displayed message on RxM. This is nearly unavoidable, but it's detection is possible during periodical manual audits where logfiles are compared. Additionally, any AI or word substitution instruction must be transmitted to RxM. It's very likely that any message displayed by malware is out of context and thus detectable; The situation can then be solved off-band.

TFC can not guarantee security of NH. This means that attacker who compromises NH can at will prevent communication between parties. However, such DoS attack does not lead to compromise of content by itself: NH absolutely never handles plaintexts or encryption keys, only ciphertexts and tags.

9.8 Data remanence

Users should store keyfiles on magnetic media to avoid the wear-levelling problem of flash-storage devices. All data should be encrypted prior to writing it on any device. Magnetic media can be reused after it has been sanitized with industry standard tools using anything from three to 35 passes. When disposing of hard drives and solid state memory, physical crushing is recommended. Hard drives can additionally also be degaussed before crushing.

9.9 Targeted attacks

Information-theoretically secure / quantum-safe symmetric ciphers, plaintexts and truly random keys that can not be exfiltrated with traditional malware makes TFC the most secure system for communication available. The system is however not unbreakable: TFC, like all other current platforms are possibly vulnerable to following sophisticated, close access operations and side channel attacks.

9.9.1 TEMPEST

TEMPEST is codename for using sensitive receivers at moderate distances to compromise the unintended electromagnetic emanations of electronic devices. EMSEC is mainly the concern of high value stationary targets known by the intelligence agency such as governmental and military facilities, large businesses, or smaller ones in key fields of interest. In TFC the main emission sources are the cables of display units and keyboards connected to TxM and RxM: the emissions of these **leak plaintext information**. To prevent this, computers and keyboards should be operated inside shielded cases that block signals from escaping. To transmit keyboard signal inside a solid case, highly sensitive piezo switches can be used. Encapsulation of display units requires metallic mesh that has the correct density to block escaping of signals emanated by the display device. The use of retro reflectors as revealed by the leaks²⁸ in EMSEC compromise might indicate that collection of passive emanations is not the preferred method any more.

9.9.2 Covert channels

Covert channels and other hidden transmitters can in theory be injected to future revisions of off-the-shelf computers. Current devices may be subjected to such compromise via firmware/OS update. This however is unlikely, since the serial interface only transmits when messages are sent and the data diode greatly increases the required voltage and reduces the frequency range in which covert channel can be hidden to. This means the existence of covert channel is detectable by mediocre measuring equipment such as oscilloscopes or logic- and spectrum analysers²⁹. In case TxM OS could be compromised during installation, OTP keys provide some security through obesity: exfiltration of keyfile is much slower and louder than exfiltration of PSKs of TFC-CEV would be, and the optocoupler caps the speed at which key material can be transmitted. Careful EMSEC protection would also protect against hidden radio transmissions sent by TxM or RxM, compromised either during manufacturing, in transit³⁰ or on scene: Computers bought off-the-shelf have lower risk rating than those that are shipped to user.

9.9.3 Acoustic cryptanalysis

Most practical device to eavesdrop is the keyboard of TxM. The attack can be mitigated either by a covert listening device, the integrated microphone of the NH or the microphone or accelerometer of a smart phone³¹. Users concerned about targeted attacks should hire a professional to conduct a bug sweep. EMSEC protected keyboards with piezo switches have no mechanical parts that make audible noise and they should thus be the safest option against acoustic attacks.

²⁸https://upload.wikimedia.org/wikipedia/commons/2/2a/NSA_RAGEMASTER.jpg

²⁹https://en.wikipedia.org/wiki/Signals_intelligence#Signal_detection

³⁰<http://www.forbes.com/sites/erikkain/2013/12/29/report-nsa-intercepting-laptops-ordered-online-installing-spyware/>

³¹<http://www.hacker10.com/other-computing/how-mobile-phone-accelerometers-are-used-for-keylogging/>

10 Conclusion

This project succeeded in adding security to both the highest layer of communication encryption stack and to the lowest OSI layer - hardware. Computational and information theoretical security remove cryptanalytic attack-vectors, making unstoppable bulk collection trivial. Availability of self-buildable, open circuit design HWRNG and data diodes allows users to build trustworthy TCB platforms, effectively rendering the short-sighted undermining of industry standards useless.

The intelligence community is currently in progress of transferring from mass decryption of naive encryption protocols with subpoenas, to bulk CNE, where end-to-end encryption is broken with automated mass-exploitation of clients using backdoors and 0-day exploits. TFC was built to be secure against these types of attacks. The mass-compromise scenario of data-diode secured communication would require ubiquitously built-in covert transmitters and/or massive fleets of SIGINT drones to capture emanations of devices. Such attacks would be incredibly expensive to mitigate, thus TFC is part of solution suggested by Bruce Schneier to make surveillance expensive again.³²

Some protections against the targeted attacks were introduced in this paper, in case price of TEMPEST type mass surveillance technology would one day drop drastically. As the main concern of users today is remote mass-exploitation of clients, expected rate of EMSEC protection will remain low and law-enforcement retains it's SIGINT capability for traditional targeted surveillance and thus, balance between individual privacy and collective security is enforced by technology.

11 Copyrights

- Copyrights of the design schematic for HWRNG belong to Giorgio Vazzana and are modified and published under GNU free license and written permission of the author.
- Licences for the software and libraries used are listed in LICENCES.md files in software repositories at Github.³³.
- Copyrights for the design schematics of the RS-232 Data Diode belong to Douglas W. Jones and are published used under GNU free license and used in this document accordingly.
- This document and manual³⁴ are released under GNU Free Documentation License 1.3
- Tx.py, NH.py, Rx.py, tfc{OTP,CEV}installer.py, In.py and genKey.py are part of the TFC application, which is free software: You can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version. TFC is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details. For a copy of the GNU General Public License, see <http://www.gnu.org/licenses/>.

³²<https://www.schneier.com/news-156.html>

³³<https://creativecommons.org/publicdomain/zero/1.0/>

³⁴<http://cs.helsinki.fi/u/oottela/tfc-manual.pdf>

12 Appendix A: Summary of TFC key features

12.1 TxM

getEntropy.c	Collects the entropy from GPIO and stores it into file
deskew.c	Removes bias from the entropy file
toBytes.c	Converts binary numbers to bytes
genKey.py	Generates keyfiles by controlling the C-programs
Tx.py	Padds, encrypts and signs sent files and messages Optional hiding of meta-data via trickle connection Additional pseudo-random intervals between consecutive packets Group management and message multicasting Overwriting of used keys and keyfiles Hash-based blacklist for used keys Encrypted Rx.py management commands Keyfile exhaust warnings and swap command Quick screen clearing and exiting against shoulder-surfing
Ent	Evaluates the quality of entropy during keyfile generation
Dieharder	Evaluates the quality of keyfile with series of tests

12.2 RxM

Rx.py	Authenticates, decrypts and/or displays/stores messages and files. Nicknames for contacts Opt-in conversation logging Encrypted management commands from Tx.py One-command keyfile swap Drops all packets the MAC of which fails Drops all replayed packets
-------	---

12.3 NH

NH.py	Relays data between TxM, RxM and IM client
-------	--

12.4 Other software

tfc{CEV,OTP}installer.py	Installs the TFC suite to desired platform
--------------------------	--

13 Appendix B: List of terms and abbreviations

AES	Advanced Encryption Standard (Rijndael)
AKE	Authenticated Key Exchange
CA	Certificate Authority
CNE	Computer Network Exploitation
DHE	Diffie-Hellman-Merkle, ephemeral key exchange
DSA	Digital Signature Algorithm
EMSEC	Emissions Security
GPIO	General-Purpose input/output
HWRNG	Hardware Random Number Generator
IC	Integrated circuit
IM	Instant messaging
IPC	Inter-process communication
ISP	Internet service provider
LED	Light-Emitting Diode
MAC	Message authentication code
MITM	Man-in-the-middle attack
mpOTR	Multi-party Off-The-Record protocol
NH	Network Handler
NSA	National Security Agency
OSI	Open Systems Interconnection
OTP	One-time-pad encryption
OTR	Off-The-Record protocol
PGP	Pretty Good Privacy
PRBG	Pseudo-random bit generator
PRISM	NSA's mass electronic surveillance data mining program
PRNG	Pseudo-random number generator
PSK	Pre-shared key
RF	Radio-frequency
RNG	Random number generator
RS-232	Recommended Standard 232, Serial interface
RSA	Rivest-Shamir-Adleman public key encryption algorithm
RxM	Receiver module
SHA	Secure Hashing Algorithm
SIGINT	Signals Intelligence
TAO	Tailored Access Operations
TCB	Trusted Computing Base
TCP	Transmission Control Protocol
TEMPEST	Code word for compromising emanations of electronic devices
TFC	Tinfoil Chat
TLA	Three-Letter-Agency (FBI, NSA etc.)
TLS	Transport Layer Security protocol
TPM	Trusted Platform Module
TxM	Transmitter module
UDHR	Universal Declaration of Human Rights
XOR	The boolean exclusive-or. Denoted by symbol \oplus